



# Approved notation for developing pseudocode

## Approved notation for developing pseudocode

When developing pseudocode teachers must use the symbols below, which are those used in mathematics.

This information should be distributed to candidates as close as possible to the commencement of teaching of the course. This notation sheet will be available to candidates during the external examinations.

Conventions	<p>Variable names are all capitals, for example, CITY</p> <p>Pseudocode keywords are lower case, for example, loop, if ...</p> <p>Method names are mixed case, for example, getRecord</p> <p>Methods are invoked using the "dot notation" used in Java, C++, C#, and similar languages, for example, BIGARRAY.binarySearch( 27 )</p>
Variable names	<p>These will be provided and comments // used, for example:</p> <p>N = 5 // the number of items in the array</p> <p>SCOREHISTORY.getExam( NUM ) // get the student's score on exam NUM</p>
Assigning a value to a variable	<p>Values will be assigned using = , for example:</p> <p>N = 5 // indicates the array has 5 data items</p> <p>VALUE[0] = 7 // assigns the first data item in the array a value of 7</p>
Output of information	<p>Output—this term is sufficient to indicate the data is output to a printer, screen, for example:</p> <p>output COUNT // display the count on the screen</p>

Symbol	Definition	Examples	
=	is equal to / assignment	X = 4, X = K	If x = 4...
≠	Is not equal to	X ≠ 4	
>	is greater than	X > 4	if X > 4 then
>=	is greater than or equal to	X >= 6	loop while X >= 6
<	is less than	VALUE[Y] < 7	loop until VALUE[Y] < 7
<=	is less than or equal to	VALUE[] <=12	if VALUE[Y] <= 12 then
AND	logical AND	A AND B	if X < 7 AND Y > 2 then
OR	logical OR	A OR B	if X < 7 OR Y > 2 then
NOT	logical NOT	NOT A	if NOT X = 7 then
mod	modulo	15 mod 7 = 1	if VALUE[Y] mod 7 = 0 then
div	integer part of quotient	15 div 7 = 2	if VALUE[Y] div 7 = 2 then

Operation	Flowchart example	Pseudocode example
<b>sequential operations</b>	<pre> graph TD     Start(( )) --&gt; Task1[perform task1]     Task1 --&gt; Task2[perform task2]     Task2 --&gt; End(( )) </pre>	<p>.</p> <p>.</p> <p>.</p> <p>perform task1</p> <p>perform task2</p> <p>.</p> <p>.</p>
<b>conditional operations</b>	<pre> graph TD     Start(( )) --&gt; Decision{MAX &gt; 0}     Decision -- YES --&gt; Task1[output "positive"]     Decision -- NO --&gt; Task2[output "negative"]     Task1 --&gt; End(( ))     Task2 --&gt; End </pre>	<p>.</p> <p>.</p> <p>.</p> <p>if MAX &gt; 0 then</p> <p>    output "positive"</p> <p>else</p> <p>    output "negative"</p> <p>end if</p> <p>.</p> <p>.</p>
<b>while-loop</b>	<pre> graph TD     Start(( )) --&gt; Decision{COUNT &lt; 15}     Decision -- YES --&gt; Task1[COUNT = COUNT + 1]     Task1 --&gt; Decision     Decision -- N --&gt; End(( )) </pre>	<p>.</p> <p>.</p> <p>.</p> <p>loop while COUNT &lt; 15</p> <p>    COUNT = COUNT + 1</p> <p>end loop</p> <p>.</p> <p>.</p>
<b>from/to-loop</b>	<pre> graph TD     Start(( )) --&gt; Task1[COUNT = 0]     Task1 --&gt; Task2[SUM = SUM + COUNT]     Task2 --&gt; Task3[COUNT = COUNT + 1]     Task3 --&gt; Decision{COUNT &gt; 5}     Decision -- YES --&gt; End(( ))     Decision -- NO --&gt; Task2 </pre>	<p>.</p> <p>.</p> <p>.</p> <p>loop COUNT from 0 to 5</p> <p>    SUM = SUM + COUNT</p> <p>end loop</p> <p>.</p> <p>.</p> <p>.</p>